

**Figure 1**

AUS9-2000-0587-US1
Sheet 1 of 6

Template	Slot 0	Slot 1	Slot 2
00	M-unit	I-unit	I-unit
01	M-unit	I-unit	I-unit
02	M-unit	I-unit	I-unit
03	M-unit	I-unit	I-unit
04	M-unit	L-unit	X-unit
05	M-unit	L-unit	X-unit
06			
07			
08	M-unit	M-unit	I-unit
09	M-unit	M-unit	I-unit
0A	M-unit	M-unit	I-unit
0B	M-unit	M-unit	I-unit
0C	M-unit	F-unit	I-unit
0D	M-unit	F-unit	I-unit
0E	M-unit	M-unit	F-unit
0F	M-unit	M-unit	F-unit
10	M-unit	I-unit	B-unit
11	M-unit	I-unit	B-unit
12	M-unit	B-unit	B-unit
13	M-unit	B-unit	B-unit
14			
15			
16	B-unit	B-unit	B-unit
17	B-unit	B-unit	B-unit
18	M-unit	M-unit	B-unit
19	M-unit	M-unit	B-unit
1A			
1B			
1C	M-unit	F-unit	B-unit
1D	M-unit	F-unit	B-unit
1E			
1F			

AUS9-2000-0587-US1
Sheet 2 of 6

[illegible]

For each instruction in the group:

```
case TypeA:
    TypesA++;
    TypesMIA++;
    break;
```

```
case TypeI:
    TypesI++;
    TypesMIA++;
    break;
```

```
case TypeF:
    TypesF++;
    break;
```

```
default:
    error;
```

```
/*-----*/
| Test for the a previous incomplete bundle
/*-----*/
```

```
if despersion window is large {
```

```
Template = M_MI;  
take-M;  
take-I;  
goto StoreBundle;
```

```

if remainder = 0 then: {
  if TypesI > 0 AND TypesF < TypesF-units AND TypesM+TypesA <

```

```

goto MakeM_MI;
else
    goto MakeMFB;

```

AUS9-2000-0587-US1
Sheet 3 of 6

```

    }
    if remainder = 1 then: {
        if (TypesI > 0 OR TypesA > 0) AND TypesF < TypesF-units AND
TypesM+TypesA < bundle count then:
            goto MakeM_MI;
        else
            goto MakeMFB;
    }
    /* remainder = 2 */
    if (TypesI > 0 OR TypesA > 0) AND TypesF < TypesF-units AND
TypesM+TypesA >= bundle count then:
        goto MakeM_MI;

```

MakeMFB:

```

    Template = MFB_;
    nop;
    nopb;
    goto StoreBundle;

```

---+/*

```

| INCOMPLETE equals MI_I
+-----

```

---*/

```

} else {
    INCOMPLETE = 0;
    if despersal window is large {

```

MakeMI_I:

```

    Template = MI_I;
    take-I;
    goto StoreBundle;

```

```

    }
    remainder = size of instruction group % 3
    if remainder = 2 then:
        goto MakeMIB;

```

```

    if remainder = 0 then: {
        if TypesI > 0 AND TypesF < TypesF-units
AND TypesM+TypesA < bundle count then: goto MakeMI_I;
        else goto MakeMIB;
    }

```

```

    /* remainder = 1 */
    if (TypesI > 0 OR TypesA > 0) AND TypesF < TypesF-units
AND TypesM+TypesA < bundle count then: goto MakeMI_I;

```

MakeMIB:

```

    Template = MIB_;
    nopB;
    goto StoreBundle;

```

}

}

While TypesALL > 0 { // while instructions remain in group

Figure 3B

AUS9-2000-0587-US1

Sheet 4 of 6

```

if TypesALL is equal to TypesMIA {
  if TypesALL > 3 {
    if TypesM > TypesI+TypesA then: Template = MMI; take-M;take-M;take-I
    else Template = MII; take-M;take-I;take-I
    goto StoreBundle;
  }
  if TypesALL = 3 and this is the first bundle of the group {
    if TypesI > 1 then: Template = MII; take-M;take-I;take-I
    else Template = MMI; take-M;take-M;take-I
    goto StoreBundle;
  }
  if TypesALL = 2 OR TypesALL = 1 and TypesI = 1{
    if TypesM = 2 then: Template = MMF; take-M;take-M;nop; goto
StoreBundle;
    else INCOMPLETE = MI_I take-M; take-I; goto TopOfGroup
  }
  /* TypesALL = 1 */
  INCOMPLETE = M_MI take-M; goto TopOfGroup
}
if TypesLX > 0 then: Template= MLX; take-M; take-LX; goto StoreBundle;

if TypesB > 0 AND TypesALL-TypesB < 3 then: {
  if typesF > 0 then: {
    if TypesF+TypesI = 2 then: Template= MFI nop;takeF;take-I; goto
StoreBundle;
    else Template=MFB take-M;takeF;take-B; goto StoreBundle;
  } else if TypesI > 0 {
    if TypesI = 2 then: Template= MII nop;takeF;take-I; goto StoreBundle;
    else Template=MIB take-M;take-I;take-B; goto StoreBundle;
  } else if TypesM = 2 then: Template = MMB take-M;take-M;take-B; goto
StoreBundle;
  else if TypesALL-TypesB = 2 then: Template = MIB take-M;take-I;take-B;
goto StoreBundle;
  else if TypesB = 1 then: Template = MFB take-M;takeF;take-B; goto
StoreBundle;
  else if there are 2 TypesB instructions or 1 non-TypesB: Template = MBB
take-M;take-B;take-B; goto StoreBundle;
  else Template = BBB take-B;take-B;take-B; goto StoreBundle;
  }
  /* TypesF > 0 */
  if TypesALL = 3 AND TypesM = 2 then: Template = MMF take-M;take-M takeF;
goto StoreBundle;
  else Template = MFI take-M;takeF take-I; goto StoreBundle;
}
storeBundle:
  if TypesALL = 0 then: insert stop bit in Template;
  build bundle in code buffer;
  if TypesALL = 0 then: goto MainLoop;
  else goto TopOfGroup;
}
DONE
  if INCOMPLETE is not zero then: {
    if INCOMPLETE = M_MI then: Template = MFB; nop; nopB; goto StoreBundle;
    else Template = MIB; nopB; goto StoreBundle;
  }
}

```

Figure 3C

AUS9-2000-0587-US1

Sheet 5 of 6

AUS9-2000-0587-US1
Sheet 6 of 6

